REMARKS/ARGUMENTS

By this paper, Applicant responds to the Office Action of September 30, 2005 and respectfully requests reconsideration of the application. The shortened statutory period runs through December 30, 2005. Accordingly, this response is timely.

Claims 1-81 are now pending, a total of 81 claims. Claims 1, 2, 24, 34, 46, 52 and 70 are independent. Of the independent claims, claims 2, 24, 34, and 46 have not been allowed over the art, however, the Office Action is also insufficient to raise any rejection of these claims.

I. Double Patenting

A. Claims 1-51

The "double patenting" rejection of paragraph 2 is not understood.

1. Did the Examiner Intend to Raise a Rejection?

The "mirror image" issue in the '401 application was withdrawn. 09/425,401, Notice of Allowance of 8/9/2005. Is the issue in this application a mere typographical error?

2. If so, What Ground? What Claims? Where is the Analysis Requred by MPEP § 804(B)(1)?

Because no Office Action has ever set out a meaningful analysis, procedurally no rejection exists, and substantively no response is possible. As noted in the accompanying Request For Withdrawal of Finality of Office Action, no Office Action has ever clearly answered any of the following questions:

- Are claims 1-51 are subject to "same invention" or "obviousness-type" double patenting?
- Which claims are involved? Of claims 1-51, at most thirty can be double patenting "twins" (either same invention or obviousness-type) of the thirty claims of the '401 application. Which thirty?
- What one particular claim is thought to correspond to what one particular claim of the '401 application, on which the Examiner thinks the best double-patenting case might arise? If obviousness-type double patenting is involved, MPEP § 804(B)(1) instructs that when making an obvious-type double patenting analysis, the following showings must be presented:

  > (A) Determine the scope and content of a patent claim [singular] and the prior art relative to a claim [singular] in the application at issue;

> (B) Determine the differences between the scope and content of the patent claim [singular] and the prior art as determined in (A) and the claim [singular] in the application at issue;

Because no Office Action has ever set out the necessary analysis, and none has never even identified a particular claim of this application that might correspond to a particular claim of the '401 application, let alone made showings (A) and (B), it is impossible to determine whether the Examiner is right or wrong, and if he's right, how to respond. Procedurally, no rejection exists.

3. What is the Basis for Stating that Certain Claim Limitations are Identical or Obvious to Claims in the '401 Application?

Applicant's paper of August 20, 2004 showed that no double patenting rejection could possibly exist between claims 1-51 of this application and the claims of the '401 application, as follows:

> Fourth, Applicant further notes that no double patenting rejection could be raised. For example, each of the independent claims of this application recites one or more of the following limitations:
>
> - recording profile information concerning the execution of the program, the profile information recording of the address of the last byte of at least one instruction (claims 1, 2 and 24)
> - ... the program being coded in an instruction set in which an interpretation of an instruction depends on a processor mode not expressed in the binary representation of the instruction; [and] recording profile information ... efficiently tailored to annotate the profiled binary code with sufficient processor mode information to resolve mode-dependency in the binary coding. (claims 1, 34 and 46).
>
> In contrast, none of the independent claims of application serial no. 09/425,401 recite either of these limitations.
>
> Applicant submits that "a clear line of demarcation" has been maintained between the claims of the two applications as requested by the Office Action. No double-patenting rejection is warranted.

The subsequent Office Actions have failed to "answer all material traversed" as required by MPEP § 707.07(f), except a bald statement that these two limitations are "inherent." However, there bas been no "basis in fact and/or technical reasoning ... that the allegedly inherent characteristic necessarily flows," MPEP § 2112, from the claims of the '401 patent. Without

that, there is no showing of inherency. And without that showing, procedurally, no rejection exists.

No rejection has been stated with sufficient specificity to determine whether applicant should respond with argument, whether claims should be cancelled, or whether a terminal disclaimer should be filed. Until the Examiner's view is clearly stated, applicant cannot respond.

B. Claims 52-81

The double patenting rejection of paragraph 3, relating to claims 52-81, is acknowledged. Good grounds exist for retaining the claims in more than one application.

First, until the Examiner clearly articulates a complete analysis of claims 1-51, or acknowledges that no double patenting issue exists, and the double patenting issues relating to claims 1-51 are resolved, it is impossible to determine whether the better course is to cancel claims 1-30 from the '401 application or to cancel claims 52-81 from this application.

Second, both the public and this applicant are entitled to a clear record, and that clear record can only be developed while the claims remain pending in both applications.

It is acknowledged that these claims are properly rejected, but the resolution of this issue must be deferred until the issue relating to claims 1-51 is resolved.

II. Claims 2-33

Claim 2 recites as follows:

> 2. A method, comprising:
>
> executing a program on a computer;
>
> recording in a memory of the computer profile information concerning the execution of the program, the profile information recording **the address of the last byte of at least one multi-byte instruction** executed by the computer during a profiled interval of the execution.

A. The Office Action is Procedurally Inadequate to Raise Any Rejection

The Office Action asserts that the "last byte" language is "inherent" in Heisch '033. As Applicant observed in the paper of July 12, 2005:

> There is no reason to believe that Heisch '033 records "the address of the last byte." Conventional profiling systems record program counter values, which indicate the address of the first byte of instructions profiled [including the first byte of the last instruction profiled]. The Office Action indicates no reason to

> believe that Heisch departs from this conventional practice, let alone 'necessarily' does so.

The Examiner takes no issue with the observation, and apparently agrees that a conventional reading is that Heisch '033 stores only the address of the first byte of any instruction profiled. With that concession, any claim of inherency evaporates.

The Office Action goes off on two irrelevant and legally impermissible tangents.

First, the Office Action suggests that Heisch '033 may be "considered" to possibly record the last byte of instructions. That is not inherency: "The fact that a certain result or characteristic **may** occur or be present in the prior art is not sufficient to establish the inherency .... In relying upon the theory of inherency, the examiner must provide a basis in fact and/or technical reasoning ... that the allegedly inherent characteristic **necessarily** flows from the teachings of the applied prior art." MPEP § 2112 (underline in original). Inherency requires a showing that there is no other possible reading of the reference. Because the Office Action does not contest that the "first byte" possibility exists, there is no inherency.

Second, the current Office Action points to Figs. 3, 4, and 8. It is impermissible to rely on figures for "quantitative values" unless the quantitative values are supported in the specification. MPEP § 2125 reads as follows (quotations and citations omitted):

> **2125 Drawings as Prior Art**
>
> Drawings and pictures can anticipate claims if they clearly show the structure which is claimed. However, the picture must show all the claimed structural features and how they are put together...
>
> When the reference does not disclose that the drawings are to scale and is silent as to dimensions, arguments based on measurement of the drawing features are of little value. See *Hockerson-Halberstadt, Inc. v. Avia Group Int'l*, 222 F.3d 951, 956, 55 USPQ2d 1487, 1491 (Fed. Cir. 2000) (The disclosure gave no indication that the drawings were drawn to scale. "[I]t is well established that patent drawings do not define the precise proportions of the elements and may not be relied on to show particular sizes if the specification is completely silent on the issue.").

MPEP § 2125 also refers to *In re Wright*, 569 F.2d 1124, 1127, 193 USPQ 332, 335 (CCPA 1977), which reads as follows (emphasis added):

> Absent any written description in the specification of quantitative values, arguments based on measurement of a drawing are of little value.

No rejection may rely solely on figures to establish the quantitative difference between the address of the first byte and the address of the last byte of a given instruction. Heisch's figures may not be relied on for either an explicit-disclosure or inherency-based rejection.

The Office Action is procedurally insufficient to raise any rejection whatsoever.

B. The Technological Analysis of the Office Action is Faulty

The Office Action states that Heisch '033 shows "in fig. 3, the first and last bytes are recorded." The Office Action is wrong. At col. 2, lines 47-50, Heisch '033 states that his preferred embodiment relates to the PowerPC. As is well-known in the art, PowerPC instructions are 4 bytes long, and all have their first byte on a 4-byte aligned boundary. Both columns of Fig. 3 shows addresses that increment by 4, and are 4-byte aligned. Fig. 3 clearly shows only the first byte addresses of instructions. "Instructions" are conventionally designated by the address of their first byte, and the Office Action indicates no reason to believe that Heisch '033 is any exception.

Claim 2 recites "profile information recording **the address of the last byte of at least one multi-byte instruction.**" Heisch '033 shows no such thing. Claim 2 is patentable.

C. Claim 24 and Dependent Claims

Independent claim 24 recites similar language, and is not rejected for reasons similar to, those discussed above

Dependent claims 3-15, 17-20, 22 and 25-31 are patentable with independent claims 2 and 24 discussed above. In addition, the dependent claims recite additional features that further distinguish the art.

III. Claims 34 and 46

Claim 34 recites as follows:

> 34. A method, comprising:
>
> executing a program on a computer, **without the program having been compiled for profiled execution**, the program being **coded in an instruction set in which an interpretation of an instruction depends on a processor mode not expressed in the binary representation of the instruction**;
>
> recording in a memory of the computer profile information describing an interval of the program's execution and processor mode during the profiled

interval of the program, the profile information having a data structural form efficiently tailored to annotate **the profiled binary code with sufficient processor mode information to resolve mode-dependency in the binary coding**.

A. The Heisch '033 Reference and "Trace Scheduling"

Heisch '033 teaches that he is using "trace directed" techniques, col. 2, lines 31-37 and col. 4, line 53. "Trace directed" is a term of art, originally coined by Joseph Fisher, "Trace Scheduling, A Technique for Global Microcode Compaction, IEEE Transactions on Computers C-30(7):478-490 (July 1981), and J. R. Ellis, Bulldog: A Compiler for VLIW Architectures, Technical Report YALEU/DCS/RR-364, Yale University, Department of Computer Science, reprinted by the MIT Press (1985).

"Trace directed" optimization and execution is a 3-step process, as shown in the top right slide of Exhibit A (http://www.lems.brown.edu/~iris/en291s9-04/lectures/291-17-trace-scheduling.pdf, hereinafter "Brown slides"). First, a program is compiled in a profiling mode (Brown slides, "During initial compilation, 'extra code' can be added to a program to generate profiling statistics when the program is executed").[1] The profiled mode includes code at every point in the program where a branch instruction changes control to either its destination or to its fall-through successor. This profiling code counts the number of times that execution follows each particular path. Second, the profiled version of the program is executed with a number of workload inputs that reflect the typical data that are expected for the program (Brown slides, "A profile is associated with a particular program input."). During this execution, path counts are counted by the profiling code. Third, the program is recompiled, using the path counts to reorder basic blocks together into "straight line" segments that will execute efficiently (Brown slides, "Recompile the program using the profile to guide optimization choices").

As another example, Multiflow Computer, the company founded based on Dr. Fisher's and Dr. Ellis' ideas, did its profiling in the compiler. (Personal conversation of 11/27/2005 with

[1] It may be the case that some embodiments of trace directed techniques have used a separate program to post-process a binary to insert profiling instrumentation code. This post-processor is part of the compiler, just as the cpp pre-processor is part of a C compiler.

Dr. Geoff Lowney, a former senior engineer at Multiflow, now an Intel Fellow, one of the 50 senior-most engineers at Intel.)

For an example, consider this little example program:

```
while (A) {
        B;
        if (C) then
                D;
        else
                E;
        F;
}
```

The compiler will generate a version with code inserted to capture profiling information, as follows, where "A_to_B++" means "add 1 to the count for the flow from A to B":

```
while (A) {
        A_to_B ++
        B;
        if (C) then
                C_to_D ++;
                D;
        } else {
                C_to_E ++;
                E;
        }
        F;
        F_to_A ++;
}
```

Assume the profiled execution (from step 1) on the sample workloads (in step 2) determines that C is usually true, and therefore E is executed more often than D. On recompilation (step 3), the program will be reordered as follows, and compiled in this form:

```
        go to A1;

B1:     B;
        if (not C) then go to D1;
        E;
F1:     F;
A1:     if A go to B1;

D1:     D;
        goto F1;
```

Note that if C is usually true, the loop from B to A will usually be executed in straight-line fashion. The only branch instruction that is usually taken (and thus, the only disruption to the fetch order of instructions and their smooth flow down the pipeline) is the one at the bottom of the loop.

The indicated portions of Heisch '033 contain nothing to suggest that Heisch does anything other than traditional trace profiling, in which a program's binary code has the instrumentation and counters for profiled execution compiled into it.

B. The Office Action Errs By Failing to Make Either A Showing of Explicit or Inherent Disclosure

There are only two ways to show anticipation: explicit teaching in a reference, or inherency. If no explicit teaching can be identified, then an Office Action must make a showing of inherency, including "basis in fact and/or technical reasoning ... that the allegedly inherent characteristic necessarily flows." MPEP § 2112, underline in original. "Probabilities or possibilities" that might be "considered" are not permissible.

The September 2005 Office Action concedes, page 3, lines 18-20, that Heisch '033 has no explicit disclosure of the claim language "without the program having been compiled for profiled execution.

The Office Action makes no attempt to show inherency by showing that the inherent material "necessarily flows," and that all other possibilities are excluded. Indeed, the Office Action concedes that Heisch '033 could be read to use a program that has been "compiled for profiled execution," as discussed in Applicant's July 2005 paper, and in § III.A, above.

The Office Action expressly concedes that Heisch '033 contains no explicit or inherent support for a rejection, and instead "considers" that Heisch '033 might depart from traditional trace directed techniques based solely on Heisch's silence. Mere silence of a reference is never a proper basis on which to raise a rejection.

The September Office Action makes no attempt to show either explicit or inherent disclosure. No rejection of claim 34 exists.

C. The Technological Analysis in the Office Action is Faulty

A "selected workload" has nothing to do with "instructions," let alone "interpretation of an instruction depends on a processor mode" or "sufficient processor mode information to resolve mode-dependency." A "selected workload" includes the "constraints" that control the execution of a software program, primarily the "program input:"

> A workload is typically in the form of program input, ... Some programs behave almost exactly the same regardless of the program input set (workload). However, other programs behave radically different based on the workload. Thus, a particular program application may run in a certain manner, given a specific workload, i.e. specific program input. However, the same program application may run radically different given another set of program inputs. Thus, ideally the program application will be optimized for the workload on which it is to be run.

Col. 14, lines 15-30, esp. lines 26. Heisch's "workload" has nothing to do with "processor mode." *See also* Brown Slides ("A profile is associated with a particular program input")

The "processor mode" language of claim 34 has not been shown to correspond to Heisch '033.

D. The Office Action is Incomplete, by Failing to Identify an Instruction Set with Instructions Whose Interpretation Depends on a "Processor Mode"

The Office Action asserts that Heisch '033 has a "a processor mode not expressed in the binary representation of the instruction," page 3, lines 18, but indicates no portion or feature of Heisch '033 that corresponds to such a "processor mode." Neither Office Action has ever designated a portion of a reference and explained the pertinence, to the claim language "coded in an instruction set in which an interpretation of an instruction depends on a processor mode not

expressed in the binary representation of the instruction." What instruction? What processor mode?

Applicant cannot guess at the Office Action's position. No rejection exists, and no response is possible.

E. The Office Action Errs By Disregarding Claim Limitations

The Office Action makes no attempt to compare the language "the profile information having a data structural form efficiently tailored to annotate the profiled binary code with sufficient processor mode information to resolve mode-dependency in the binary coding" to any reference. Instead, the Office Action states that this language is disregarded as "merely a desired result." The Office Action cites no authority that supports disregard of claim limitations. The MPEP instructs otherwise: even in a preamble, "statements ... reciting ... intended use must be evaluated..." MPEP § 2111.02. "Desired results" in a claim body are always given patentable weight. *Ex parte Mead*, http://www.uspto.gov/web/offices/dcom/bpai/decisions/fd001501.pdf at 2-3 (Bd. pat. App. & Interf. 2002); *Ex parte Touhseant*, http://www.uspto.gov/web/offices/dcom/bpai/decisions/fd990203.pdf at 9 (Bd. Pat. App. & Interf. 2002). The courts and Board have noted that language similar to that in claim 34 may be quite broad, but not so broad that it may be totally disregarded.

The Office Action errs by stating "nothing in the claim indicates that the feature actually occurs, just that the system is capable of performing the desired feature." That is not correct. The claim affirmatively recites that the profile information has "a data structural form" with certain attributes, not that it is merely "capable of" having that form.

The Office Action makes no attempt to "designate" the portions of Heisch '033 relied on, and "explain the pertinence" as required by 37 C.F.R. § 1.104(c)(2). Instead, the Action simply refers to a large paragraph, and invites Applicant to guess at the relevance. Applicant is unable to discern how that paragraph of Heisch '033 might be considered to produce profiling information "having a data structural form efficiently tailored to annotate the profiled binary code with sufficient processor mode information to resolve mode-dependency in the binary

coding." Until a reasonably complete and precise element-by-element showing is provided, no rejection exists.

F. Claims 35-40, 44, 46-49

Claim 46 recites similar language, and is similarly not rejected. The claims dependent thereon, 35-45 and 47-55, are similarly not rejected.

IV. Conclusion

After six years, this application has not yet received a complete first examination. Applicant again requests that any future rejection include all showings required by the relevant provisions of the MPEP. Every claim limitation should be addressed. The relevant portions of any reference must be "designated as nearly as practicable." The pertinence should be clearly explained. 37 C.F.R. § 1.104(c)(2). All showings of "inherency" should be properly supported. The Examiner should not make up new rules out of thin air, for example, about claim limitations that may be disregarded, or double patenting. If claim limitations are to be disregarded, or claims are rejected "in bulk" without limitation-by-limitation comparison, the Examiner should cite some authority that states the rule being applied.

Applicant respectfully submits that the claims are in condition for allowance. Applicant requests that the application be passed to issue in due course. The Examiner is urged to telephone Applicant's undersigned counsel at the number noted below if it will advance the prosecution of this application, or with any suggestion to resolve any condition that would impede allowance. In the event that any extension of time is required, Applicant petitions for that extension of time required to make this response timely. Kindly charge any additional fee, or credit any surplus, to Deposit Account No. 23-2405, Order No. 114596-07-4014.

Respectfully submitted,

WILLKIE FARR & GALLAGHER LLP

Dated: November 30, 2005

By: _____

David E. Boundy
Registration No. 36,461

WILLKIE FARR & GALLAGHER LLP
787 Seventh Ave.
New York, New York 10019
(212) 728-8757
(212) 728-9757 Fax

BROWN

Advanced Computer Architecture Lecture 17: Trace Scheduling

Prof. R. Iris Bahar

EN291-S01

October 18, 2004

Reading Assignments: *Superscalar Microprocessor Design, Mike Johnson Chapter 11.1, 11.4 (optional), 11.5*

Article Assignment: *"When Caches Aren't Enough: Data Prefetching Techniques", by Vander Wiel, Lilja (available on webpage)*

BROWN

Profile Driven Optimizations

- Wrong optimization choices can be costly!
  How do you determine dynamic information during compilation?
- During initial compilation, "extra code" can be added to a program to generate profiling statistics when the program is executed
- Execution Profile, e.g.
  - how many times is a basic block executed
  - how often is a branch taken vs. not taken
- Recompile the program using the profile to guide optimization choices
- A profile is associated with a particular program input
  ⇒ *may not work well on all executions*

BROWN

Trace Scheduling *[Josh Fisher]*

- Generate multi-basic block traces based on profiling information
  - find the most often executed control path
- List schedule a trace at a time
  - optimize the execution of the trace *(common case)*
  - fix any problem with off-trace paths as necessary *(infrequently executed)*
- Good for very biased and predictable branching behavior
- Trace scheduling engendered the VLIW architecture innovation and was implemented in the Multiflow TRACE compiler, which provided the basis for superscalar compilation techniques

BROWN

Trace Scheduling Overview

- Trace Selection
  - select seed (the highest frequency basic block)
  - extend trace (along the highest frequency edges)
    forward (successor of the last block of the trace)
    backward (predecessor of the first block of the trace)
  - don't cross loop back edge
  - bound max_trace_length heuristically
- Trace Scheduling
  - build data precedence graph for a whole trace
  - perform list scheduling and allocate registers
  - add compensation code to maintain semantic correctness
- Speculative Code Motion (upward)
  - Move an instruction above branches if safe